软件设计与开发

Issac

复旦大学

微电子科学与工程系

2018.6

施蕊	蔡俊哲	王渝
15307130017	15307130076	15307130258

目录

1 游戏概述

Issac是S.C.W于2018年06月发行的一款2D平面角色扮演、动作冒险RPG类的独立游戏，主角Issac将在有着能够提升能力的道具与特殊技能的半RPG世界中闯荡。

游戏胜利条件：战胜迷宫阶层主蝇王。

游戏操作：WASD控制Issac的移动，上下左右控制Issac的射击，E键放下炸弹。在开始界面按Enter进入游戏模式，开始界面按D进入Debug模式。

我们使用了cocos2dx游戏引擎，使用C++语言分别在Windows和macOS平台构建了整个Issac游戏。

关键技术

MVC 框架

单例模式

AOP 编程模型

BFS 最短路径算法

Prim 算法随机迷宫

数据抽象与面向对象技术

类接口技术

Lambda 表达式作为回调函数

工厂模式和装饰模式

物理引擎

节点树渲染

2 系统需求说明

2.1 系统开发工具

Windows: Visual Studio 2017 + Cocos2dx 3.16

maxOS: Xcode 9.2 + Cocos2dx 3.16 + Sketch 48.2

2.2 系统运行环境

Windows 10 1803

macOS Sierra 10.12.6

2.3 依赖

glfw 3.1.2

3 系统设计

3.1 软件组织框架

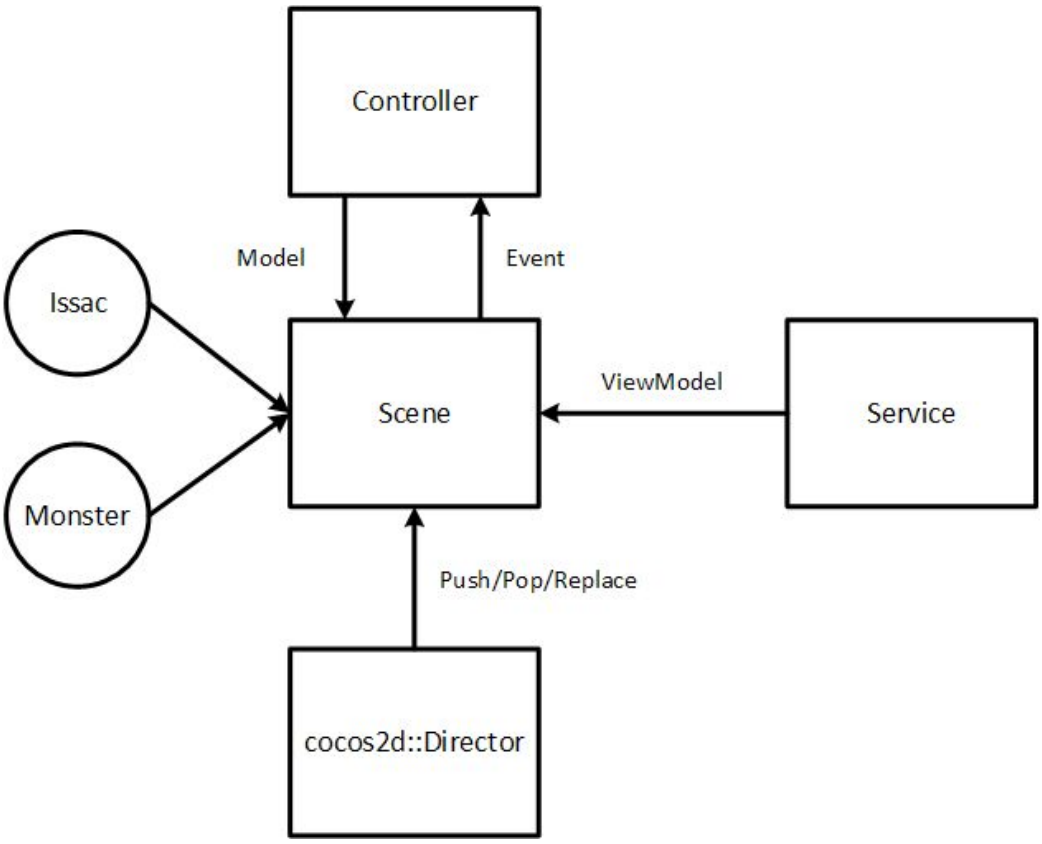


图1.框架图

cocos2d::Director控制Scene的切换，Scene与Controller结合形成MVC框架。Scene中包含Issac、Monster等物体。Scene与Controller通过Event和Model交换信息。Service使用单例和AOP编程模型。Scene调用Service方法获得ViewModel，根据ViewModel中的信息，条件渲染Scene中的物体。

3.2 软件运行逻辑与细节

1. 软件运行后进入欢迎界面，用户输入 Enter 后，进入主菜单。
2. 在主菜单用户可以通过上下方向键控制选择光标，输入 Enter 后游戏开始，主菜单销毁。
3. 游戏主界面为空房间，房间地面背景上有基本操作提示，用户输入 WASD 分别控制 issac 上下左右运动，输入方向键↑↓←→分别控制 issac 发射 Tear，输入 E 控制 issac 放置炸弹。
4. 所有的房间均有门通向其他房间，issac 进入其他房间后，当前房间销毁，新房间生成，但 service 保存所有房间信息，issac 返回后，原始房间信息还原。
5. 所有房间地面均有地面石头和物品。

6. 所有房间均有边界，所有对象只能在边界中运动。当 issac 进入带有怪物的房间后，房间门会立即关闭，issac 无法逃出。只有当所有怪物死亡后，房间门才会打开。
7. boss 房间带有 boss 血条。
8. 当 issac 击败 boss 后，显示获胜界面，用户可以通过输入 Esc 或空格选择退出到主菜单或在游戏界面重新开始。
9. issac 自身游戏属性有：生命值、移动速度、攻击力、Tear 射速、 Tear 射程、炸弹数、子弹击中石头和墙壁后是否可以回弹、是否可以飞行。
10. 房间顶层有 HUD 面板，显示用户生命值、可放置的炸弹数，如用户没有炸弹仍然放置，则炸弹数闪烁以通知用户。有半透明小地图（透明度可调），显示用户当前房间和相邻房间（可区分普通房间、宝藏房间和 boss 房间），issac 当前所处房间始终居中。
11. 在任一游戏运行界面若用户输入 Esc 键，游戏界面暂停（音乐仍然播放）并显示暂停菜单，暂停菜单上显示issac 游戏属性信息（实时更新）。用户可以输入方向键↑↓控制暂停界面光标，输入 Enter 跳转到响应界面。若用户选择 resume，则游戏继续进行。若用户选择 exit game，则游戏退出，当前游戏销毁并显示主菜单。若用户选择 options，则显示设置界面。
12. issac 可拾取地面物品，拾取物品后 issac 游戏属性变化。issac 与怪物或怪物的 Tear 碰撞，或在炸弹爆炸范围内时，受到伤害，生命值减少（不同伤害生命值减少不同），issac 角色闪烁以提示用户，并播放受伤音效。怪物在死亡后显示死亡动画和地面特效。issac 可飞行后，可以在石头上运动。
13. 炸弹放置后闪烁一定时间，然后显示爆炸效果，在一定爆炸范围内的怪物或 issac 受到伤害。爆炸后地面显示效果。
14. 在设置界面，用户输入方向键↑↓控制设置界面光标，输入方向键←→调节对应设置项数值。可调设置项有：音量、音效、小地图透明度。
15. 主菜单、游戏界面、游戏失败界面背景音乐各不相同。

3.3 Listener

Listener 下共有两个文件，IMainSceneListener.h 和IRoomSceneListener.h。Listener 声明 Controller 所要用到的函数接口。

1. on_key_pressed
   响应用户按下键盘
2. on_key_released
   响应用户释放键盘

3.4 Controller

Controller 负责初始化场景，初始化音乐引擎，设置事件响应，处理用户的输入并更新 model。Controller 与 View 没有直接数据交换。Controller 继承 Scene 和 Listener 的接口。

1. MainSceneController
2. RoomSceneController

3.5 Model

Model是Controller向Scene传递信息的载体。

1. MainSceneModel中包含与主界面菜单相关的数据。
2. RoomSceneModel中包含玩家移动和攻击的数据和一些内嵌菜单状态的相关数据。

3.6 ViewModel

ViewModel由Service创建，传递给Scene，然后Scene根据ViewModel条件渲染。

3.6.1 RoomViewModel

static RoomViewModel createRoomViewModel(int roomType, bool visited, int barrierType, bool item_taken)是RoomViewModel的主要功能，创建一个房间。它由RoomService给出房间编号和地形编号，然后根据此生成对应房间的信息。

房间编号	房间内容
roomType:0	初始房间
roomType:1~7	怪物房间。1:全Fatty，2:全Fly，3:全Gaper，4:全Spider，5:Spider+FattyFire，6:Fly+FlyFire，7:Fly+GaperFire
roomType:8~20	宝藏房间。宝藏房号20是测试用，会将所有宝藏放在房间里
roomType:21+	代表Boss房,21:Boss-flyDaddy
地形编号	地形内容
barrierType:0	空地
barrierType:1	在中央小石头排成十字形

barrierType:2	四块大石头

RoomViewModel包含需要渲染一个房间的全部信息。包括：

- 房间门的位置和贴图
- 地面贴图和墙贴图
- 是否已经访问过
- 房间的编号
- 房间内具体信息：人为的将房间可移动部分划分成一张13*7的棋盘。每个格子内保存这个格子的内容。内容编号：0表示空地，1表示小石头，2表示大石头，3表示玩家，4~19表示怪物的各种类别，20~30表示宝藏

3.6.2 PlayerViewModel

PlayerViewModel存放玩家的各项属性。包括：

- 生命值
- 攻击值
- 移动速度
- 眼泪速度
- 眼泪射程
- 眼泪间隔时间
- 碰撞半径
- 质量
- 飞行和弹跳
- 炸弹数量

Issac也就是玩家的属性是全局都可以访问修改，这也是将Issac属性从Issac类中抽取到Service里的初衷。

3.6.3 MiniMapViewModel

MiniMapViewModel包含正确显示小地图的信息。主要有两个成员：

- 小地图二维数组信息
- 透明度

3.7 Service

Service使用单例和AOP编程模型，有RoomService和PlayerService两个主要类。

RoomService负责管理房间，PlayerService负责管理玩家的状态。首先，游戏启动时，RoomService内部维护一个map，生成地图，这个地图可以是随机生成的迷宫，也可以是手动设计的地图。map里的数据结构是地图服务的核心，所有辅助函数都会访问这个结构的成员。其成员如下：

类型	成员	注释
int	current_room_id	当前房间ID
int	left_room_id	左边房间ID，如果不存在则为0
int	up_room_id	上边房间ID，如果不存在则为0
int	right_room_id	右边房间ID，如果不存在则为0
int	down_room_id	下边房间ID，如果不存在则为0
bool	visited	房间是否被访问过
bool	item_taken	房间里的物品是否被拿走
int	current_room_type	房间类型，具体有 0：初始房间 1~7：怪物房间。1:全Fatty,2:全Fly ,3:全Gaper ,4:全Spider,5:Spider+FattyFire,6:Fly+FlyFire,7:Fly+GaperFire 8~20：宝藏房间,宝藏房号20是测试用，会将所有宝藏放在房间里 21+：代表Boss房,21:Boss-flyDaddy
int	current_barrier_type	石头的生成形状，具体是： 0：没有石头 1：十字形 2：田字形

RoomService会根据这个信息生成RoomViewModel，Scene中拿到这个RoomViewModel就可以绘制出正确的房间。

3.7.1 Debug模式固定地图

为了测试游戏的各项功能，我们还做了一套固定的地图，其结构如下：

图2.测试地图

房间的类型和怪物类型和贴图可以简单由room_type决定。在测试的时候可以方便设置每一个的类型，单独测试每一个房间。

3.7.2 正式模式的复杂迷宫

在游戏启动正式模式时（enter进入），系统不会调用Debug模式的固定地图，而会使用另一个初始函数产生10*10的迷宫，当前初始地位于迷宫左下角，且出生地边上一定有一个宝藏房间；Boss位于地图的右上角，打败boss即可通关游戏。

迷宫地形通过prim算法随机生成。此算法随机生成的迷宫保证从出生地一定有路能够到达终点boss房间，且生成的迷宫十分美观。迷宫为每一个能够走的房间配给一个独特的RoomID。

在迷宫生成后，通过循环遍历每个房间来将迷宫转换成上述定义的map类型，房间四周的RoomID可通过访问迷宫得到。房间初始是否访问=false。房间的地形和类型完全随机，三种地形等概率出现，房间类型有5%的几率为宝藏房间，有95%几率为怪物房间。

3.7.3 小地图生成及音乐

小地图的生成算法是两层遍历，先遍历当前房间的所有方向，然后再遍历四个方向房间各自的四个方向房间，如果，房间没有访问到是不会进行二次遍历（一次遍历始终进行），最终会生成5×5的二维数组数据。在Scene中会根据这个二维数组加载正确的贴图到正确的地方。

RoomService还管理了背景音乐的大小和小地图的透明度。

RoomService提供了许多辅助方法，方便Scene调用Service。

PlayerService比RoomService简单，主要负责保存玩家属性，另外也提供了许多辅助函数方便调用。

3.8 Scene

Scene 负责绘制场景和场景内所有物体的操作。

3.8.1 MainScene主菜单的场景

cocos2dx 中在绘制场景时通过 zorder 对各个 Node 进行按序遍历，视觉上产生图层。

MainScene 成员如下：

类型	名称	说明
cocos2d::Sprite *	bg	欢迎界面和主菜单
cocos2d::Sprite *	selector	主菜单光标
MainSceneModel	model	MainScene 的 model
IMainSceneListener *	listener_	设置事件监听
int	viewflag_	存储当前主界面视图 id
cocos2d::Size	size	存储当前窗口尺寸
int	menuflag_	存储光标选中的菜单项 id
int	selector_init_x_	存储光标初始位置横坐标
int	selector_init_y_	存储光标初始位置纵坐标

MainScene 较为简单，负责欢迎界面和主菜单的更新，Zorder 在此省略。该 Scene 只有一个 Sprite，包含欢迎界面和主菜单，该 Sprite 结构如下：

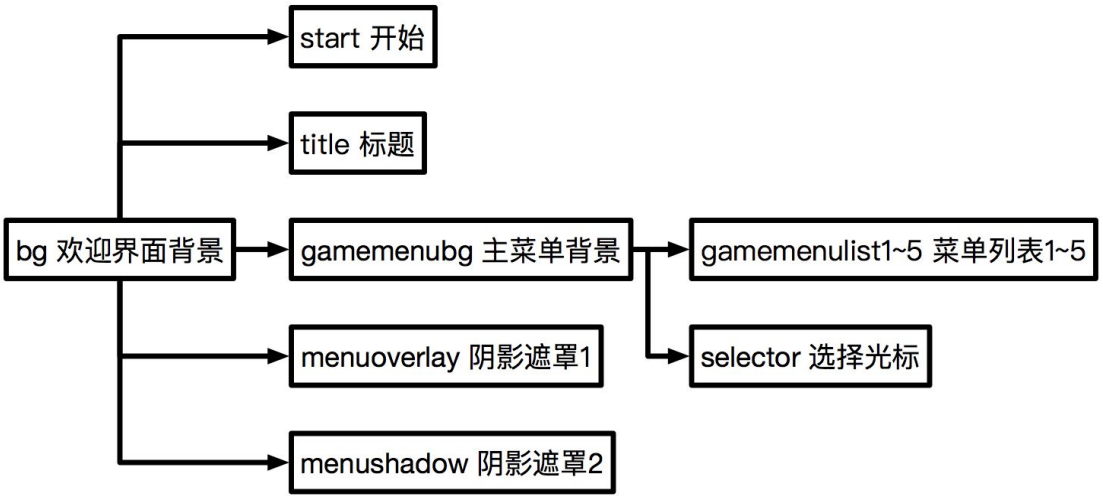


图3. MainScene 结构图

在该类的 update 函数中，通过获得 viewflag_信息，判断当前视图是否需要移动（在欢迎界面和主菜单之间切换）。

3.8.2 RoomScene游戏场景

RoomScene成员和成员函数如下：

类型	名称	说明	Zorder 或容器内项目 Zorder
RoomSceneModel	model	RoomScene 的 model	无
Issac *	player	玩家控制的角色	3
cocos2d::Sprite *	room_piece1	房间地面左上角	0
cocos2d::Sprite *	room_piece2	房间地面右上角	0
cocos2d::Sprite *	room_piece3	房间地面左下角	0
cocos2d::Sprite *	room_piece4	房间地面右下角	0
cocos2d::Sprite *	controls	初始房间地面上的操作提示	0
cocos2d::Sprite *	shading	房间光线阴影	2
cocos2d::Sprite *	overlay	房间全局光线遮罩	6
cocos2d::Sprite *	edgeSpace	房间的周围边界（物理引擎检测用）	默认值
cocos2d::Sprite *	tearSprite	issac 发射的 Tear	默认值

cocos2d::Sprite *	pausescreen	暂停菜单	7
cocos2d::Sprite *	optionscreen	设置菜单	7
cocos2d::Sprite *	deadscreen	死亡菜单	9
cocos2d::Sprite *	winscreen	获胜菜单	10
cocos2d::SpriteFrame *	fullheart	healthbar 的元素	无
cocos2d::SpriteFrame *	halfheart	healthbar 的元素	无
cocos2d::Sprite *	bosshealthbar	boss 血条	6
cocos2d::Sprite *	minimap	小地图	5
cocos2d::Sprite *	healthbar	玩家血量 HUD	8
cocos2d::Sprite *	hud_bomb	玩家炸弹 HUD 图标	8
cocos2d::Label *	Count_bomb	玩家炸弹数标签	8
cocos2d::Sprite *	bomb	炸弹	3
cocos2d::Sprite *	bombraduis	炸弹弹坑	2
cocos2d::Sprite *	blood_pool	血迹	2
cocos2d::Sprite *	ghost_sprite	issac 死亡后的幽灵	3
cocos2d::Sprite *	deadbody	issac 死亡的尸体	3
cocos2d::Sprite *	collectible_streak	拾取物品提示	6
cocos2d::Sprite *	temp_sprite	引导 Tear 播放销毁动画的1x1透明贴图	3
cocos2d::Vector<Monster*>	monsters_	容器，存放怪物指针	3
cocos2d::Vector<Tear*>	tears_	容器，存放 Tear 指针	默认值
cocos2d::Vector<Stone*>	stones_	容器，存放石头指针	2
cocos2d::Vector<Door*>	doors_	容器，存放门指针	1
cocos2d::Vector<Collectable*>	collectables_	容器，存放拾取物品指针	3
int	prev_bomb_num	前一帧的炸弹数	无
IRoomSceneListener *	listener_	设置事件监听	无
MiniMapViewModel	mini_map_vm_	小地图 Viewmodel	无

RoomViewModel	room_vm_	房间 Viewmodel	无

成员函数	功能说明
void set_event_listener(IRoomSceneListener *listener);	设置事件监听
void update(float delta) override;	更新
void fire(float dt);	player 发射 Tear
bool onContactBegin(PhysicsContact& contact);	物理碰撞监听
void build_frame_cache() const;	构造房间需要的帧缓存和动画缓存，包括：Tear 帧缓存，MonsterTear 帧缓存，血量 HUD 全心和半心的帧缓存，设置菜单中状态条贴图，暂停菜单中 player 状态的状态条贴图，房间弹坑、炸弹爆炸和血迹的帧缓存和动画缓存

3.8.2.1 RoomScene中的房间生成

RoomScene的init()函数是场景初始化。

根据RoomID到RoomService中取得当前房间的生成信息RoomViewModel。根据RoomViewModel中的13*7棋盘，在每一个空格处放上对应的物体（怪物、玩家、石头、门），并生成无形边界。

同时，RoomScene要使用initWithPhysics进行物理引擎物理世界的初始化。

3.8.2.2 RoomScene中的物体控制

场景中需要控制的物体为子弹、怪物和玩家，这三种物体的控制在update()中按帧进行计算。

子弹：对所有场景中的子弹，判断其存在时间是否为0，如果为0子弹消失并播放子弹消失动画；不为0存在时间计数器-1。

怪物：对场景所有的怪物，判断其血量是否为0，如果为0怪物消失并播放怪物消失动画；不为0则依次调用怪物的移动策略，射击策略和生怪物策略。

玩家：判断其血量是否为0，为0则播放死亡动画和失败界面，不为0则根据键盘监听进行移动和射击。

3.8.2.3 RoomScene中的场景绘制

pausescreen 结构如下：

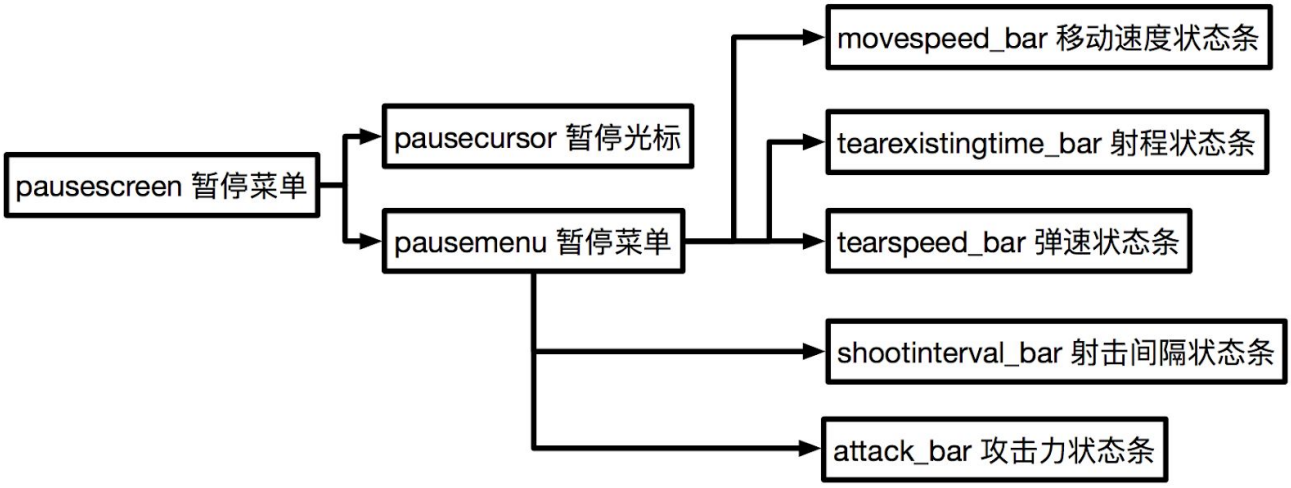
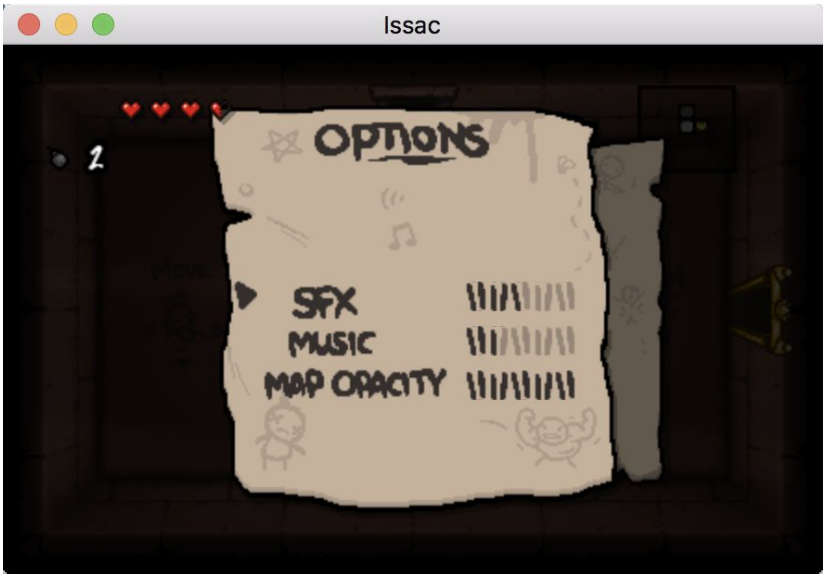


图4. pausescreen绘制结构图

optionscreen 结构如下：



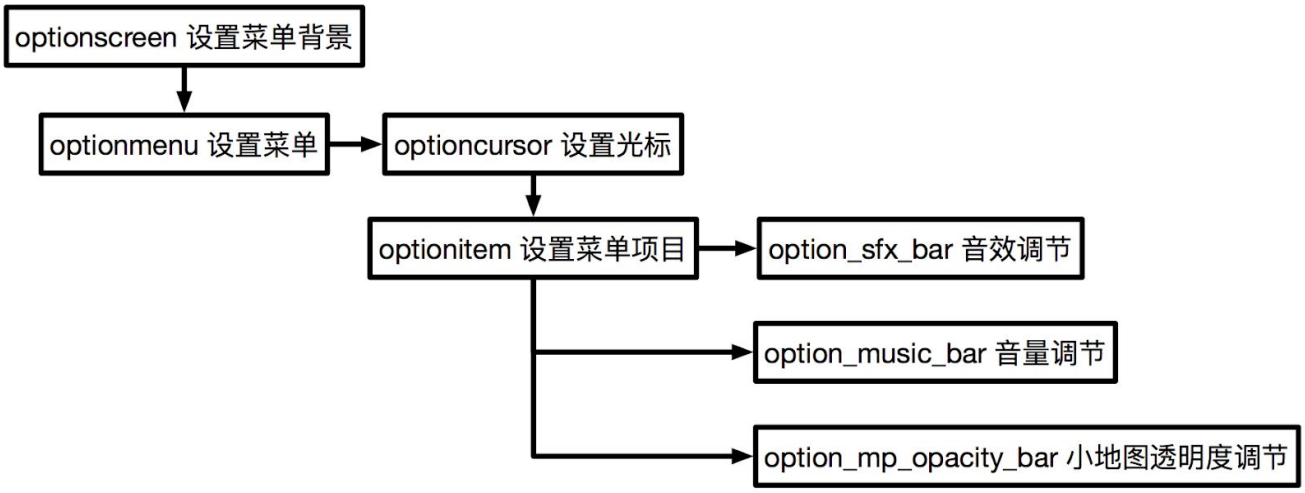
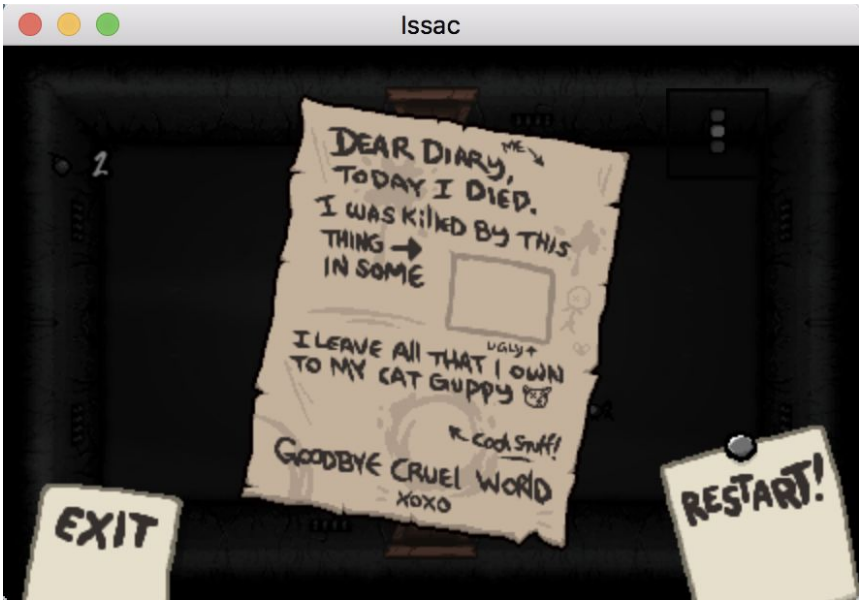


图5. optionscreen绘制结构图

deadscreen 结构如下：



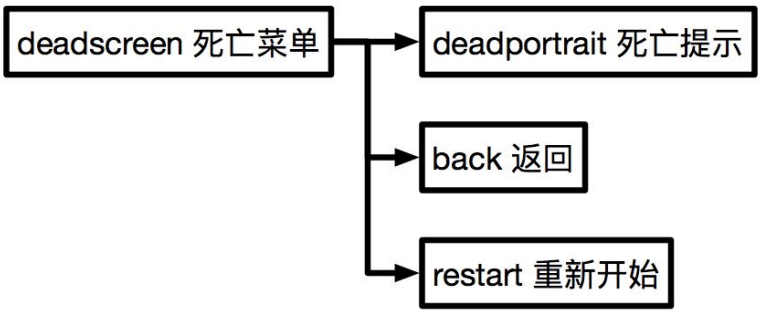


图6. deadscreen绘制结构图

winscreen 结构如下：

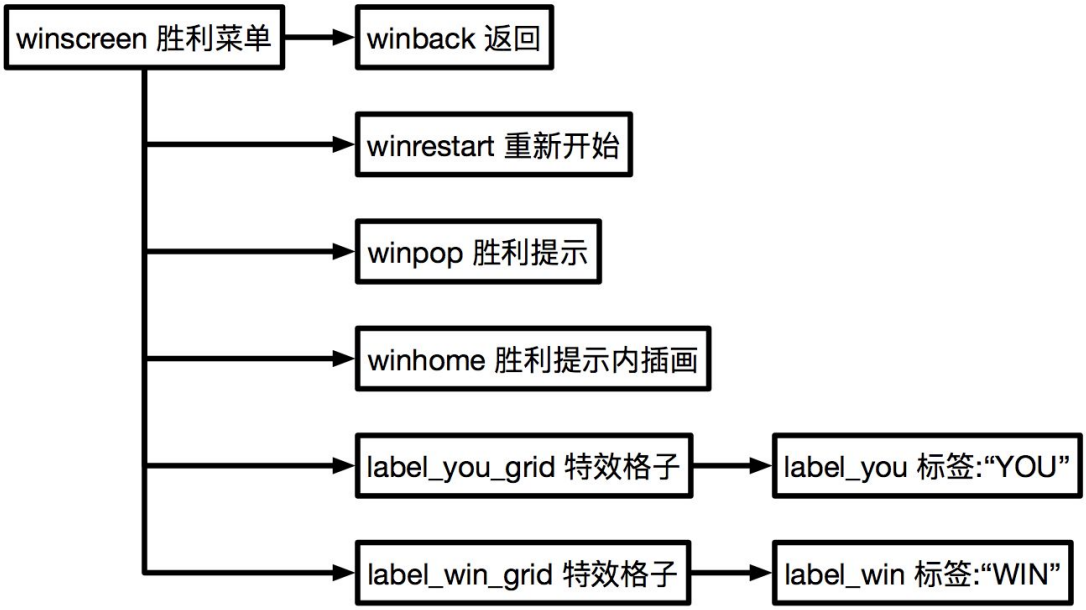


图7. winscreen绘制结构图

在该类的 update 函数中，通过 model.game_stat 的值分为四个分支：

model.game_stat = 0：运行状态。每一帧进行的操作如下：

1. 首先判断前一阵暂停界面是否显示，如果显示则设为不可见，并将标志位 model.paused_menu_generated_flag 置 1。
2. 通过 model.bomb 判断炸弹是否放下，如果是，则先将model.bomb置否，然后判断 player 的炸弹数：如果大于0，则player 炸弹数减1，并在 player 当前位置按序生成炸弹、炸弹闪烁效果、炸弹爆炸效果和弹坑。如果等于0，则只使炸弹计数闪烁以提醒玩家。
3. 通过 player->getBombNum()>prev_bomb_num 即当前玩家炸弹数是否大于前一帧玩家炸弹数，判断玩家是否捡到炸弹：如果是，则更新炸弹数。

4. 对 tear_ 容器中的成员进行遍历计算判断 tear 存在时间：如果 tear 存在时间为0，则播放销毁音效和动画，并从容器中将该 tear 移除。否则每一帧 tear 存在时间减 1。
5. 如果 player 已经死亡，monsters_ 容器中 monster 全部静止不动。
6. 判断 monsters_ 容器中 monster 的生命值：如果小于0则表明死亡，生成死亡动画、血迹。否则更新怪物动作：移动、开火、boss 生小苍蝇。
7. 如果 player 能飞，则设置对应物理对象属性
8. player 移动，player 静止时替换身体贴图为静止状态
9. player 受伤时进行无敌时间倒计时
10. 如果 player 血量为0，通过 dead_ani_generated 判断死亡动画是否播放过：如果否，则播放死亡动画，生成幽灵，并播放死亡音效
11. 通过 scheduler 对 player 的射击行为进行相应
12. 每一帧刷新玩家的血量
13. 通过 room_vm_.getVisited() && !door_removed && !doors_.empty() 判断房间门是否已经访问过，如果是则遍历 doors_ 容器打开门。通过 monsters_.empty() && !door_removed && !doors_.empty() 判断当前房间怪物是否全部死亡，如果是则遍历 doors_ 容器打开门
14. 如果当前房间是 boss 房而且 monsters_ 容器为空，则获胜，game_stat跳转到3

model.game_stat = 1：暂停状态。由 model.pause_menu_generated_flag 判断是否生成过生成过死亡菜单。若未生成过，则显示暂停菜单并将标志位置1。停止发射函数 fire 的 scheduler。更新 issac 的状态属性。通过 model.paused_menu_cursor 更新选项光标。通过判断 model.option_display 是否为 1 来显示设置菜单。在设置菜单中，更新光标和设置项数值。

model.game_stat = 2：死亡状态。由 model.dead_menu_generated_flag 判断是否生成过生成过死亡菜单。若未生成过，则显示死亡菜单并将标志位置1。停止所有怪物动作和 scheduler，停止发射函数 fire 的 scheduler。播放死亡状态背景音乐。

model.game_stat = 3：获胜状态。由 model.win_generated_flag 判断是否生成过生成过获胜菜单。若未生成过，则显示获胜菜单并将标志位置1。停止 issac 动作。

3.9 Character

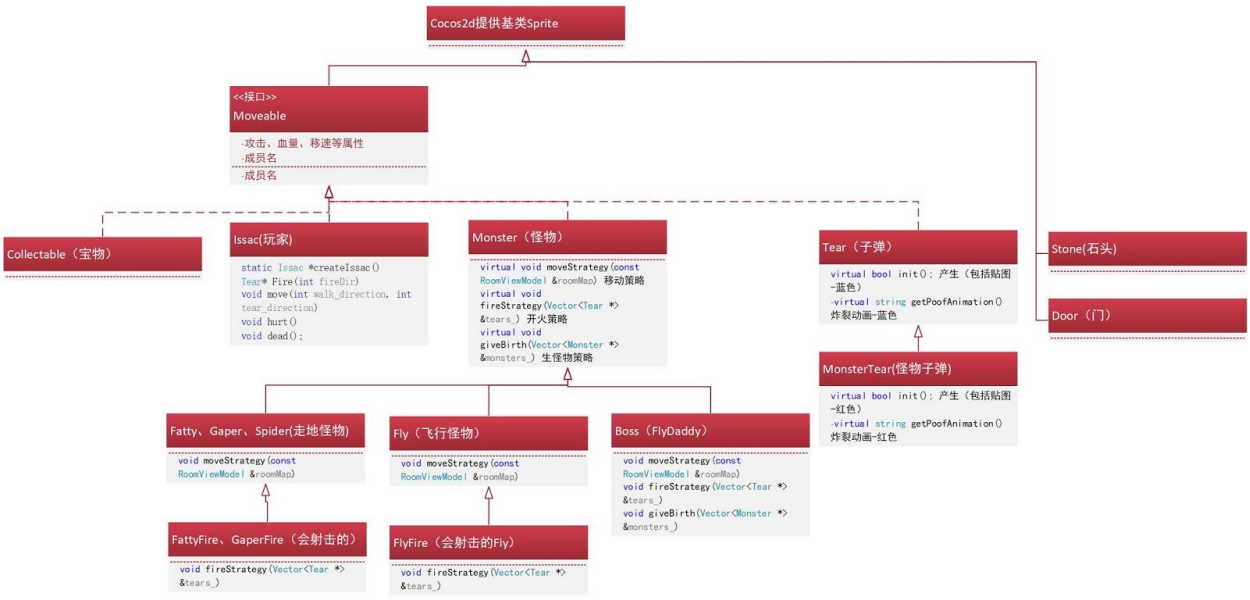


图8. Character 类图

3.9.1 基类/接口类Moveable

Moveable是所有可动物体的基类，与此同时，他也作为一个接口类，提供了物体各种属性的接口（可动物体和宝物的接口）。他提供的属性和属性get、set接口有：

类型	属性	属性说明
double	moveSpeed	移速
double	radiusSize	身体大小
double	bodyMass	质量
int	invincibleTime	受伤后无敌时间
double	tearSpeed	子弹速度
int	tearExistTime	子弹射程
double	shootInterval	射击间隔
bool	enFly	是否能飞
bool	enBounce	子弹是否反弹
bool	bombNum	炸弹数量

int	health	血量
double	attack	攻击
int	colClog	移动阻塞

3.9.2 玩家类Issac

主要函数：

static Issac *createIssac(); 类的产生和初始化

void move(int walk_direction, int tear_direction) 物理世界下的移动

Tear* Fire(int fireDir) const 射击

void hurt(); 受伤动画

void dead(); 死亡动画

3.9.3 怪物类Monster

Monster是所有怪物的基类，包括4种小怪(胖子Fatty，苍蝇Fly，普通人Gaper以及蜘蛛Spider)，其中胖子Fatty，苍蝇Fly，普通人Gaper还有同样长相，会射击的子类怪物FattyFire，FlyFire和GaperFire，以及Boss蝇王FlyDaddy。

Monster及其子类主要重写三个虚函数，来实现所有AI和互动。

virtual void moveStrategy(const RoomViewModel &roomMap) 怪物移动策略

virtual void fireStrategy(Vector<Tear *> &tears_) 怪物开火策略

virtual void giveBirth(Vector<Monster *> &monsters_) 怪物生孩子策略

移动策略名	移动策略描述
随机移动	随机向任意方向行走
简单移动	径直向玩家方向移动，如果中间有障碍物不会绕过
冲刺+停顿移动	随机向玩家方向高速冲刺或者原地不动
最短路径移动	利用BFS计算最短路径并沿着最短路径移动，会绕过障碍物
最短路径移动+变速	在最短路径移动的基础上，距离玩家一定距离内会移动速度变快

射击策略名	射击策略描述
指向射击	向玩家位置射出一个子弹

十字射击	向四周射出十字形的4个子弹
X字射击	向四周射出X字形的4个子弹

普通怪物没有生孩子策略，一共七组怪物是以上策略的组合。

怪物名	移动策略	开火策略	图片
Fatty	最短路径移动	-	
FattyFire	随机移动	十字射击	
Fly	简单移动（飞行怪物不会碰撞石头）	-	
FlyFire	简单移动（飞行怪物不会碰撞石头）	指向射击	
Gaper	最短路径移动+变速	-	
GaperFire	最短路径移动+变速	X字射击	
Spider	冲刺+停顿移动	-	

Boss拥有不一样的策略，三种策略都是独有的，并且公用一个冷却时间。

Boss	策略	描述
FlyDaddy 	移动策略	1/5几率向玩家冲刺，其余以缓慢速度在房间中移动
	射击策略	如果没有冲刺，1/4几率向四周8个方向发射3轮间隔短的子弹
	生怪物策略	如果没有冲刺，也没有射击，1/2几率产生3个苍蝇Fly，1/2几率产生1个射击苍蝇FlyFire

3.9.4 子弹类Tear

子弹类的生成，击中后的爆炸动画和音效。

3.9.5 宝物类Collectable

根据宝物编号生成十种不同功能的宝物，有不同的贴图和拾取效果。

宝物名	宝物编号	宝物属性	宝物图片
半颗心	0	加半颗血（health+1）	
一颗心	1	加一颗血（health+2）	
蘑菇	2	加攻击0.3，加血半颗心，略微加移动速度，射速，射程	
不明药剂	3	加攻击3，扣1颗心的血，减慢移动速度	
骨头	4	加攻击1	
眼镜	5	减攻击0.7，射程几乎无限	
恶魔召唤	6	加攻击4，射程减半	
弹球	7	子弹现在反弹了	
恶魔之翼	8	人能够飞了（不和石头碰撞）	
炸弹	9	增加一个炸弹	

3.9.6 石头Stone

石头是一个静态的物体，分为可见的石头和不可见的边界石头。不可见的边界石头为房间划定了边界，框定物体只能在房间中移动。可见石头分为小石头（27*27）和大石头(54*54)，是房间内的地形，走地的物体无法通过，飞行的物体可以通过。

3.9.7 门Door

门是一个静态的物体，在房间内怪物全部死亡后，玩家能够通过门进入下一个房间。

3.10 物理引擎

房间中的物体都是在物理世界中移动的。Cocos2d提供了物理引擎，来模拟物体的移动、碰撞。物理世界中的物体分为静态物体和动态物体。场景中石头、边界、门是静态物体，玩家、怪物、子弹是动态物体。

3.10.1 物理引擎下的创建

Scene的创建：Scene需要创建一个物理世界。物理世界容纳着所有被添加进去的抽象物体，并然后将整个世界作为一个整体进行更新。物理世界控制着所有元素的相互作用。

物体的创建：物理引擎中需要为每一个对象创建一个物理世界的身体，其中包括了速度、形状、恢复力、摩擦力、密度，质量等。

3.10.2 物理引擎下的移动

决定物理世界中物体移动的主要是速度。物理世界中有固定的更新时间间隔（步进），每一次更新后，物理世界中的物体会延速度方向移动，移动距离为速度*更新时间间隔。因此，在控制物体的移动时，我们只需控制物体的速度。

在Issac和Monster类中，通过Move函数对其的速度进行控制。其控制部分在RoomScene的update()中已经讲过，现在不在继续赘述。Tear类的速度在生成的时候已经确定并且不会更改。

3.10.3 物理引擎下的碰撞

物理引擎下，物体是否碰撞和物体的类别掩码categoryBitmask和碰撞掩码collisionBitmask有关。

对于物体A和B，碰撞的条件：

(A->categoryBitmask && B->collisionBitmask) || (B->categoryBitmask && A->collisionBitmask) ==1

因此，需要设计物理的分类掩码的碰撞掩码：

种类	CategoryBitmask	碰撞规则	CollisionBitmask
玩家	0000_0000_0001(001)	所有	1000_1111_1111(8FF)
走地怪物	0000_0000_0010(002)	所有	1001_1111_1111(9FF)

飞行怪物	0000_0000_0100(004)	不和普通石头碰撞	1001_1101_1111(9DF)
子弹	0000_0000_1000(008)	所有	1111_1111_1111(FFF)
边界	0000_0001_0000(010)	不和门、无形边界石头、普通石头碰撞	1111_1000_1111(F8F)
石头	0000_0010_0000(020)	不和门、无形边界石头、普通石头、飞行怪物和飞行玩家碰撞	1010_1000_1011(A8B)
门	0000_0101_0000(050)	不和门、无形边界石头、普通石头碰撞	1111_1000_1111(F8F)
宝藏	0000_1000_0000(080)	所有	1001_1111_1111(9FF)
飞\玩家	0001_0000_0000(100)	不和普通石头碰撞	1000_1101_1111(8DF)
无敌\玩家	0010_0000_0000(200)	不和走地，飞行怪物碰撞	1000_1111_1001(8F9)
无敌飞\玩家	0100_0000_0000(400)	不和走地，飞行怪物，普通石头碰撞	1000_1101_1001(8D9)
反弹子弹	1000_0000_0000(800)	所有	1111_1111_1111(FFF)

碰撞后的状态由碰撞物体的两个参数决定：

恢复力决定反弹速度中法线分量和碰撞速度的关系。

摩擦力决定反弹速度中切线分量和碰撞速度的关系。

碰撞后的阻塞：

在怪物、玩家和任何物体碰撞后，他们就像现实世界一样会不由自主地后退，期间无法进行任何操作。为此，在程序中添加了碰撞阻塞功能，在碰撞发生后的数帧内无法对他们进行操作。

3.10.4 物理引擎下的碰撞检测

物理引擎下，如果碰撞满足检测条件，会产生一个一种由物理引擎创建的用于管理两个形状碰撞的对象Contact。为了处理碰撞，在Scene中添加监听器onContactBegin(const PhysicsContact& contact)。

碰撞检测和物体的类别掩码categoryBitmask和检测掩码contactTestBitmask有关。

对于物体A和B，如果他们已经发生碰撞，检测的条件：

(A->categoryBitmask && B->contactTestBitmask) || (B->categoryBitmask && A->contactTestBitmask) ==1

因此，需要设计检测掩码：

检测掩码	低位											高位
种类	玩家	走地怪物	飞行怪物	子弹	边界	石头	门	宝藏	飞/玩家	无敌/玩家	无敌飞/玩家	反弹子弹
玩家		1	1	1			1	1				1
走地怪物	1			1					1			1
飞行怪物	1			1					1			1
子弹	1	1	1	1	1	1	1	1	1	1	1	1
边界				1								
石头				1								
门	1			1					1	1	1	
宝藏	1			1					1	1	1	
飞\玩家		1	1	1			1	1				1
无敌\玩家				1			1	1				1
无敌飞\玩家				1			1	1				1
反弹子弹	1	1	1	1					1	1	1	1

3.10.5 监听器onContactBegin的碰撞处理

对象Contact会返回碰撞的两个类，这两个类都是接口类Moveable的子类，可用接口类Moveable对其进行操作。

（a）通过两个碰撞对象的Tag判断身份

tag=0 静止障碍物，tag=1玩家， tag=2怪物， tag=3怪物子弹，tag=4玩家子弹，tag=5,6,7,8是左、上、右、下4个门，tag=9宝藏

（b）根据身份进行操作

玩家和怪物碰撞：玩家损血，进入无敌，播放受伤动画

玩家和怪物眼泪碰撞：玩家损血，进入无敌，播放受伤动画

怪物和玩家、怪物眼泪碰撞：怪物损血

玩家和门碰撞：如果当前房间没有怪物，出房间

玩家和宝藏碰撞：拾取宝藏，增加属性

眼泪和任意碰撞：碰撞后消失

3.11 Resource组织结构

目录	子目录	功能
font		存放软件 GUI 用到的全部字体
gfx		存放软件 GUI 用到的全部贴图，该级目录下有部分常用贴图
	backdrop	存放游戏各房间的背景贴图、房间全局阴影遮罩贴图、主角高光贴图
	bosses	存放游戏 boss 贴图
	characters	存放主角 issac 贴图
	effects	存放图形特效贴图，如爆炸效果、血迹
	grid	存放房间的门、石头的贴图
	items	存放可拾取物品的贴图
	monsters	存放普通怪物的贴图
	overlays	存放每种房间的光线阴影遮罩贴图
	ui	存放所有菜单贴图
music		存放软件用到的全部背景音乐
sfx		存放软件用到的全部音乐特效

4 分工情况与 git

本项目使用git作为版本控制，托管在coding.net网站上，作为私有项目，不对外开放。

4.1 成员分工

施蕾

前期调研，键盘控制逻辑，贴图和动画，房间渲染，菜单逻辑，游戏音效

蔡俊哲

房间物体生成，物体控制，游戏逻辑，物理引擎集成，随机地图生成

王渝

服务逻辑，MVC框架，房间系统，小地图，git管理

4.2 项目统计

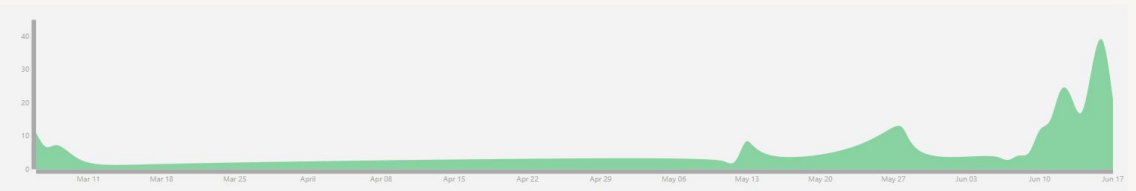
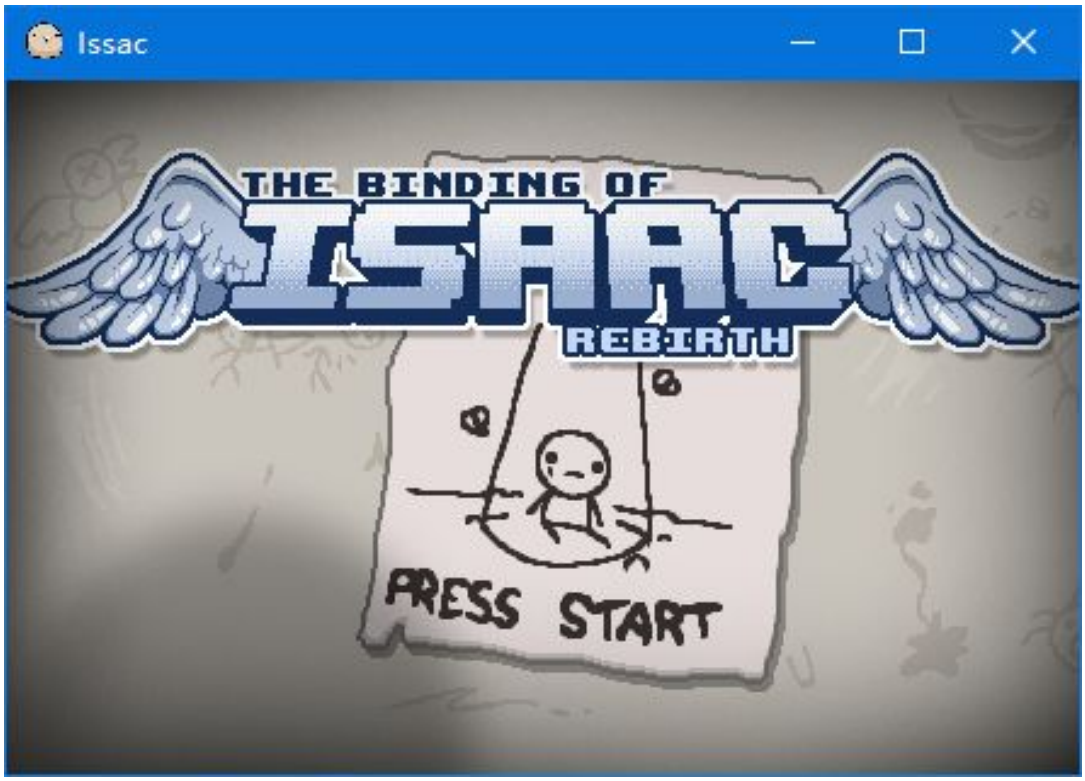


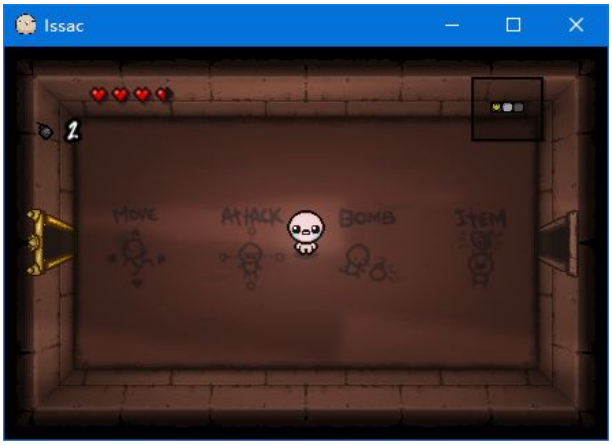
图9. 项目活跃图

总共有273个commit，代码总量2万+。

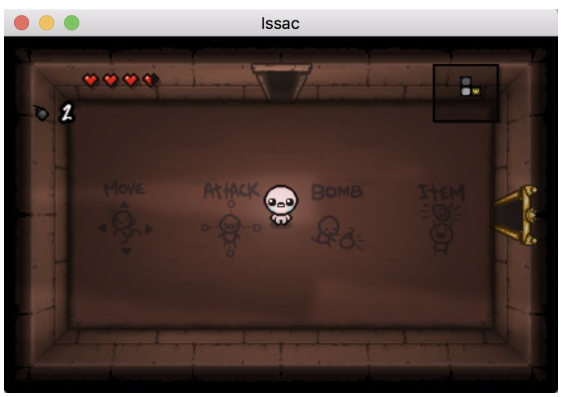
5 游戏截图

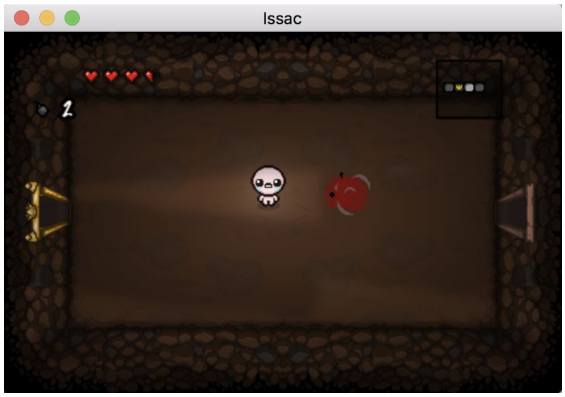
5.1 Windows下游戏运行效果





5.2 macOS 下游戏运行效果


Issac
MOVE
ATTACK
BOMB
ITEM


Issac